
Contents

1	The First Few Steps	1
1.1	What Is a Program? And What Is Programming?	1
1.2	A Python Program with Variables	3
1.2.1	The Program	4
1.2.2	Dissection of the Program	4
1.2.3	Why Not Just Use a Pocket Calculator?	5
1.2.4	Why You Must Use a Text Editor to Write Programs	6
1.2.5	Installation of Python	6
1.2.6	Write and Run Your First Program	6
1.3	A Python Program with a Library Function	7
1.4	A Python Program with Vectorization and Plotting	9
1.5	More Basic Concepts	12
1.5.1	Using Python Interactively	12
1.5.2	Arithmetics, Parentheses and Rounding Errors	13
1.5.3	Variables and Objects	13
1.5.4	Integer Division	14
1.5.5	Formatting Text and Numbers	15
1.5.6	Arrays	17
1.5.7	Plotting	18
1.5.8	Error Messages and Warnings	21
1.5.9	Input Data	23
1.5.10	Symbolic Computations	23
1.5.11	Concluding Remarks	25
1.6	Exercises	26
2	Basic Constructions	29
2.1	If Tests, Colon and Indentation	29
2.2	Functions	31
2.3	For Loops	37
2.4	While Loops	40
2.5	Lists and Tuples – Alternatives to Arrays	41
2.6	Reading from and Writing to Files	43
2.7	Exercises	44

3	Computing Integrals	55
3.1	Basic Ideas of Numerical Integration	56
3.2	The Composite Trapezoidal Rule	57
3.2.1	The General Formula	59
3.2.2	Implementation	60
3.2.3	Making a Module	62
3.2.4	Alternative Flat Special-Purpose Implementation	63
3.3	The Composite Midpoint Method	65
3.3.1	The General Formula	66
3.3.2	Implementation	67
3.3.3	Comparing the Trapezoidal and the Midpoint Methods	67
3.4	Testing	68
3.4.1	Problems with Brief Testing Procedures	68
3.4.2	Proper Test Procedures	69
3.4.3	Finite Precision of Floating-Point Numbers	71
3.4.4	Constructing Unit Tests and Writing Test Functions	72
3.5	Vectorization	76
3.6	Measuring Computational Speed	77
3.7	Double and Triple Integrals	78
3.7.1	The Midpoint Rule for a Double Integral	78
3.7.2	The Midpoint Rule for a Triple Integral	81
3.7.3	Monte Carlo Integration for Complex-Shaped Domains	84
3.8	Exercises	88
4	Solving Ordinary Differential Equations	95
4.1	Population Growth	96
4.1.1	Derivation of the Model	97
4.1.2	Numerical Solution	99
4.1.3	Programming the Forward Euler Scheme; the Special Case	102
4.1.4	Understanding the Forward Euler Method	105
4.1.5	Programming the Forward Euler Scheme; the General Case	105
4.1.6	Making the Population Growth Model More Realistic	106
4.1.7	Verification: Exact Linear Solution of the Discrete Equations	109
4.2	Spreading of Diseases	110
4.2.1	Spreading of a Flu	110
4.2.2	A Forward Euler Method for the Differential Equation System	113
4.2.3	Programming the Numerical Method; the Special Case	114
4.2.4	Outbreak or Not	116
4.2.5	Abstract Problem and Notation	116
4.2.6	Programming the Numerical Method; the General Case	117
4.2.7	Time-Restricted Immunity	119
4.2.8	Incorporating Vaccination	120
4.2.9	Discontinuous Coefficients: A Vaccination Campaign	123
4.3	Oscillating One-Dimensional Systems	124
4.3.1	Derivation of a Simple Model	124
4.3.2	Numerical Solution	126
4.3.3	Programming the Numerical Method; the Special Case	126

4.3.4	A Magic Fix of the Numerical Method	129
4.3.5	The 2nd-Order Runge-Kutta Method (or Heun's Method)	131
4.3.6	Software for Solving ODEs	132
4.3.7	The 4th-Order Runge-Kutta Method	138
4.3.8	More Effects: Damping, Nonlinearity, and External Forces	141
4.3.9	Illustration of Linear Damping	144
4.3.10	Illustration of Linear Damping with Sinusoidal Excitation	146
4.3.11	Spring-Mass System with Sliding Friction	147
4.3.12	A finite Difference Method; Undamped, Linear Case	149
4.3.13	A Finite Difference Method; Linear Damping	151
4.4	Exercises	153
5	Solving Partial Differential Equations	161
5.1	Finite Difference Methods	163
5.1.1	Reduction of a PDE to a System of ODEs	164
5.1.2	Construction of a Test Problem with Known Discrete Solution	166
5.1.3	Implementation: Forward Euler Method	166
5.1.4	Application: Heat Conduction in a Rod	168
5.1.5	Vectorization	172
5.1.6	Using Odespy to Solve the System of ODEs	173
5.1.7	Implicit Methods	174
5.2	Exercises	177
6	Solving Nonlinear Algebraic Equations	185
6.1	Brute Force Methods	186
6.1.1	Brute Force Root Finding	187
6.1.2	Brute Force Optimization	189
6.1.3	Model Problem for Algebraic Equations	190
6.2	Newton's Method	190
6.2.1	Deriving and Implementing Newton's Method	191
6.2.2	Making a More Efficient and Robust Implementation	193
6.3	The Secant Method	197
6.4	The Bisection Method	199
6.5	Rate of Convergence	201
6.6	Solving Multiple Nonlinear Algebraic Equations	203
6.6.1	Abstract Notation	203
6.6.2	Taylor Expansions for Multi-Variable Functions	204
6.6.3	Newton's Method	205
6.6.4	Implementation	205
6.7	Exercises	206
A	Getting Access to Python	209
A.1	Required Software	209
A.2	Anaconda and Spyder	211
A.2.1	Spyder on Mac	211
A.2.2	Installation of Additional Packages	211
A.3	How to Write and Run a Python Program	211

A.3.1	The Need for a Text Editor	212
A.3.2	Text Editors	212
A.3.3	Terminal Windows	212
A.3.4	Using a Plain Text Editor and a Terminal Window	213
A.3.5	Spyder	213
A.4	The SageMathCloud and Wakari Web Services	214
A.4.1	Basic Intro to SageMathCloud	214
A.4.2	Basic Intro to Wakari	215
A.4.3	Installing Your Own Python Packages	215
A.5	Writing IPython Notebooks	215
A.5.1	A Simple Program in the Notebook	216
A.5.2	Mixing Text, Mathematics, Code, and Graphics	216
References		219
Index		221

List of Exercises

Exercise 1.1: Error messages	26
Exercise 1.2: Volume of a cube	27
Exercise 1.3: Area and circumference of a circle	27
Exercise 1.4: Volumes of three cubes	27
Exercise 1.5: Average of integers	27
Exercise 1.6: Interactive computing of volume and area	27
Exercise 1.7: Peculiar results from division	28
Exercise 1.8: Update variable at command prompt	28
Exercise 1.9: Formatted print to screen	28
Exercise 1.10: Python documentation and random numbers	28
Exercise 2.1: Errors with colon, indent, etc.	44
Exercise 2.2: Compare integers a and b	45
Exercise 2.3: Functions for circumference and area of a circle	45
Exercise 2.4: Function for area of a rectangle	45
Exercise 2.5: Area of a polygon	45
Exercise 2.6: Average of integers	46
Exercise 2.7: While loop with errors	47
Exercise 2.8: Area of rectangle versus circle	47
Exercise 2.9: Find crossing points of two graphs	47
Exercise 2.10: Sort array with numbers	47
Exercise 2.11: Compute π	48
Exercise 2.12: Compute combinations of sets	48
Exercise 2.13: Frequency of random numbers	49
Exercise 2.14: Game 21	49
Exercise 2.15: Linear interpolation	49
Exercise 2.16: Test straight line requirement	50
Exercise 2.17: Fit straight line to data	50
Exercise 2.18: Fit sines to straight line	51
Exercise 2.19: Count occurrences of a string in a string	52
Exercise 3.1: Hand calculations for the trapezoidal method	88
Exercise 3.2: Hand calculations for the midpoint method	88
Exercise 3.3: Compute a simple integral	88
Exercise 3.4: Hand-calculations with sine integrals	88
Exercise 3.5: Make test functions for the midpoint method	88

Exercise 3.6: Explore rounding errors with large numbers	89
Exercise 3.7: Write test functions for $\int_0^4 \sqrt{x} dx$	89
Exercise 3.8: Rectangle methods	89
Exercise 3.9: Adaptive integration	90
Exercise 3.10: Integrating x raised to x	90
Exercise 3.11: Integrate products of sine functions	91
Exercise 3.12: Revisit fit of sines to a function	91
Exercise 3.13: Derive the trapezoidal rule for a double integral	92
Exercise 3.14: Compute the area of a triangle by Monte Carlo integration	92
Exercise 4.1: Geometric construction of the Forward Euler method	153
Exercise 4.2: Make test functions for the Forward Euler method	153
Exercise 4.3: Implement and evaluate Heun's method	153
Exercise 4.4: Find an appropriate time step; logistic model	154
Exercise 4.5: Find an appropriate time step; SIR model	154
Exercise 4.6: Model an adaptive vaccination campaign	154
Exercise 4.7: Make a SIRV model with time-limited effect of vaccination	154
Exercise 4.8: Refactor a flat program	155
Exercise 4.9: Simulate oscillations by a general ODE solver	155
Exercise 4.10: Compute the energy in oscillations	155
Exercise 4.11: Use a Backward Euler scheme for population growth	156
Exercise 4.12: Use a Crank-Nicolson scheme for population growth	156
Exercise 4.13: Understand finite differences via Taylor series	157
Exercise 4.14: Use a Backward Euler scheme for oscillations	158
Exercise 4.15: Use Heun's method for the SIR model	159
Exercise 4.16: Use Odespy to solve a simple ODE	159
Exercise 4.17: Set up a Backward Euler scheme for oscillations	159
Exercise 4.18: Set up a Forward Euler scheme for nonlinear and damped oscillations	160
Exercise 4.19: Discretize an initial condition	160
Exercise 5.1: Simulate a diffusion equation by hand	177
Exercise 5.2: Compute temperature variations in the ground	178
Exercise 5.3: Compare implicit methods	178
Exercise 5.4: Explore adaptive and implicit methods	179
Exercise 5.5: Investigate the θ rule	179
Exercise 5.6: Compute the diffusion of a Gaussian peak	180
Exercise 5.7: Vectorize a function for computing the area of a polygon	181
Exercise 5.8: Explore symmetry	181
Exercise 5.9: Compute solutions as $t \rightarrow \infty$	182
Exercise 5.10: Solve a two-point boundary value problem	183
Exercise 6.1: Understand why Newton's method can fail	206
Exercise 6.2: See if the secant method fails	207
Exercise 6.3: Understand why the bisection method cannot fail	207
Exercise 6.4: Combine the bisection method with Newton's method	207
Exercise 6.5: Write a test function for Newton's method	207
Exercise 6.6: Solve nonlinear equation for a vibrating beam	207